Amendments to the Specification:

Please replace paragraph 1 at page 7, lines 1 - 10 with the following amended paragraph:

Turning now to the attached figures, Fig. 1 depicts a conventional file system that is rooted at directory called "/" 100 and is referred to as the root directory. The root directory 100 has subdirectories "usr" 110, "tmp" 120, and "etc" 130 branching off of it. Each of these subdirectories directories 110-130 contains files 121-122 or other subdirectories 111, and 131 which contain files 132 and 133. The UNIX "ls" command executable file 112 resides at "/usr/bin" 111 and its pathname is referred to as "/usr/bin/ls" 112. When a user types the command "ls," the system consults a special environment variable that instructs the system where "ls" could possibly be located. That variable is called a PATH environment variable, and an example of a PATH environment variable is provided below.

Please replace paragraph 2 at page 8, lines 4 - 23 with the following amended paragraph:

Figs. 3 and 4 depicts a conventional UFM subsystem that receives a directory pointer and a file name to look for in that directory from NFS Server (SRV) 301. In this illustrative example, the directory is "/tmp" and the file name is "ls." To speed such frequent searches, the UFM 302 operates with virtual memory manager (VMM) 304, and uses a Name Cache 303 that contains recent successful search results. However, an entry for "/tmp/ls" is not going to exist in the name cache because no such file exists in "/tmp." The Name Cache 303 remembers only successful results, not failures. In the case of frequent failures, the Name Cache 303 is of no help, and the Name Cache 303 actually decelerates the processing system because it spends CPU cycles trying to locate something that does not exist. After failing to find "/tmp/ls" in the Name Cache 303, the system moves on and ask the FFM 305 to go look for "/tmp/ls." FFM 305 asks the Buffer Manager (BM) 306, which also operates with the VMM 304, to provide a buffer containing a disk copy of the directory contents. This step may involve disk Input/Output (I/O) 308. The FFM 305 then traverses the directory contents one file at a time until the file is found or the end of the directory has been reached. This step involves accessing the Block Special File Manager (BSFM) 307 and accessing disk Input/Output 308. In the case of a huge "/tmp" directory, the system must inspect each one of its thousands of files before all entries are

exhausted to declare failure. The problem with this search is that the system doesn't remember the results of that painful computation and it throws it away. The next time the "ls" command is invoked, the system repeats that expensive failure sequence of computations. Figure 4 is a flowchart describing the above algorithm and illustrates how conventional name caches are populated.

Please replace paragraph 3 at page 8, line 25 through page 9, line 7 with the following amended paragraph:

Figs. 5 and 6 depict the UFM subsystem of the present invention including the negative name cache that receives a directory pointer and a file name to look for in that directory. In this illustrative embodiment, the directory is "/tmp" and the file name is "ls." To speed such frequent searches, the UFM 302 uses a both a Name Cache 303 that contains recent successful search results and a Negative Name Cache 503 that contains recent unsuccessful search results. In this second illustrative embodiment, the instruction includes a request to check the Negative Name Cache 501 for an entry for "/tmp/ls". This entry is going to exist in the Negative Name Cache 503 because no such file exists in "/tmp" as established by a directory/filename search for "tmp/ls." The Negative Name Cache 503 remembers the failed attempt to locate an entry for "/tmp/ls," and the system accelerates its CPU time by immediately identifying the invalid search request and avoiding the repeated computation used by conventional systems. If not found in either Name Cache 303 or Negative Name Cache 503, the system proceeds as before to the Flat File Management (FFM) 305. If not found by the FFM 305, an entry is made to Add to Negative Name Cache 502 to speed up a later search looking for the file. Figure 6 is a flowchart describing the above algorithm and illustrates how an embodiment of the negative name cache is populated.

Please replace paragraph 2 at page 10, lines 9 - 17 with the following amended paragraph:

In another embodiment, this invention includes (a) logic to check the cache to determine if a received memory component identifier is invalid, (b) logic to add the invalid memory component identifier to the cache, such as adding a file identifier to the cache when a search in a

directory for a file name is unsuccessful, (c) logic to monitor new memory component identifiers being created and to remove an entry from the invalid list if that file identifier later becomes valid, (d) logic to monitor memory component identifiers being deleted or moved and to add an entry from the invalid list if that file identifier becomes invalid, (e) logic to manage the most frequently used invalid memory component identifiers, and (f) logic to manage the most recently used invalid memory component identifiers. Such invalid identifiers may include: 1) identifiers that are moved from a first storage location to a second storage location; 2) identifiers that are deleted; 3) identifiers that are dynamic; 4) identifiers that are renamed; and 5: identifiers selected by a user.